

# An Improvement to Levenshtein's Upper Bound on the Cardinality of Deletion Correcting Codes

Daniel Cullina

Dep. of Electrical & Computer Eng.  
Coordinated Science Laboratory  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801  
cullina@illinois.edu

Negar Kiyavash

Dep. of Industrial and Enterprise Systems Eng.  
Coordinated Science Laboratory  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801  
kiyavash@illinois.edu

**Abstract**—We consider deletion correcting codes over a  $q$ -ary alphabet. It is well known that any code capable of correcting  $s$  deletions can also correct any combination of  $s$  total insertions and deletions. To obtain asymptotic upper bounds on code size, we apply a packing argument to channels that perform different mixtures of insertions and deletions. Even though the set of codes is identical for all of these channels, the bounds that we obtain vary. Prior to this work, only the bounds corresponding to the all insertion case and the all deletion case were known. We recover these as special cases. The bound from the all deletion case, due to Levenshtein, has been the best known for more than forty five years. Our generalized bound is better than Levenshtein's bound whenever the number of deletions to be corrected is larger than the alphabet size.

## I. INTRODUCTION

Deletion channels output only a subsequence of their input while preserving the order of the transmitted symbols. Deletion channels are related to synchronization problems, a wide variety of problems in bioinformatics, and the communication of information over packet networks. This paper concerns channels that take a fixed length input string of symbols drawn from  $q$ -ary alphabet and delete a fixed number of symbols. In particular, we are interested in upper bounds on the cardinality of the largest possible  $s$ -deletion correcting codebook.

The first such upper bound is due to Levenshtein. He derived asymptotic upper and lower bounds on the sizes of binary codes for any number of deletions [5]. These bounds easily generalize to the  $q$ -ary case. He showed that the Varshamov Tenengolts (VT) codes, which had been designed to correct a single asymmetric error [10], [11], could be used to correct a single deletion. The VT codes meet the upper bound and establish its tightness in the case of a binary alphabet and a single deletion.

Since then, a wide variety of code constructions, which provide lower bounds, have been proposed for the deletion channel and other closely related channels. One recent construction uses constant Hamming weight deletion constructing codes [2]. In contrast, progress on upper bounds has been rare. Levenshtein eventually refined his original asymptotic bound (and the parallel nonbinary bound of Tenengolts) into a nonasymptotic version [7]. Kulkarni and Kiyavash recently

proved a better upper bound for an arbitrary number of deletions and any alphabet size [4].

Another line of work has attacked some related combinatorial problems. These include characterization of the sets of superstrings and substrings of any string. Levenshtein showed that the number of superstrings does not depend on the starting string [6]. He also gave upper and lower bounds on the number of substrings using the number of runs in the starting string [5]. Calabi and Hartnett gave a tight bound on the number of substrings of each length [1]. Hirschberg extended the bound to larger alphabets [3]. Swart and Ferreira gave a formula for the number of distinct substrings produced by two deletions for any starting string [9]. Liron and Langberg improved and unified existing bounds and constructed tightness examples [8]. Some of our intermediate results contribute to this area.

### A. Upper bound technique

To derive our upper bounds, we use a simple and general packing argument that can be applied to any combinatorial channel. Any combinatorial channel can be represented by a bipartite graph. Channel inputs correspond to left vertices, channel outputs correspond to right vertices, and each edge connects an input to an output that can be produced from it. If two channel inputs share a common output, they cannot both appear in the same code. The degree of an input vertex in the graph is the number of possible channel outputs for that input. If the degree of each input is at least  $r$  and there are  $N$  possible outputs, any code contains at most  $N/r$  codewords.

For a channel that makes at most  $s$  substitution errors, this argument leads to the well known Hamming bound. Any code capable of correcting  $s$  deletions is also capable of correcting any combination of  $s$  total insertions and deletions (See Lemma 2). Despite this equivalence, this packing argument produces different upper bounds for channel that perform different mixtures of insertions and deletions. Prior to this work, the bounds coming from the  $s$ -insertion channel and the  $s$ -deletion channel were known.

For the  $s$ -insertion channel, each  $q$ -ary  $n$ -symbol input has the same degree. For fixed  $q$  and  $s$ , the degree is asymptotic to  $\binom{n}{s}(q-1)^s$  (See (1)). There are  $q^{n+s}$  possible outputs, so

an  $s$ -insertion correcting code contains asymptotically at most  $q^{n+s}/\binom{n}{s}(q-1)^s$  codewords.

The  $s$ -deletion case is slightly more complicated because different inputs have different degrees. The input strings consisting of a single symbol repeated  $n$  times have only a single possible output: the string with that symbol repeated  $n-s$  times. Consequently, using the minimum degree over all of the inputs yields a worthless bound. The following argument is due to Levenshtein [5]. The average degree of an input is asymptotic to  $\left(\frac{q-1}{q}\right)^s \binom{n}{s}$  and most inputs have a degree close to that. The inputs can be divided into two classes: those with degree at least  $1-\epsilon$  times the average degree and those with smaller degree. For an appropriately chosen  $\epsilon$  that goes to zero as  $n$  goes to infinity, the vast majority of inputs fall into the former class. Call members of the former class the typical inputs. The minimum degree argument can be applied to bound the number of typical inputs that can appear in a code. There are  $q^{n-s}$  possible outputs so asymptotically at most  $q^n/\binom{n}{s}(q-1)^s$  typical inputs are in a code. We have no information about how many of the atypical inputs appear in a code, but the total number of atypical inputs is small enough to not affect the asymptotics of the upper bound.

The two bounds have the same growth rates, but the bound on deletion correcting codes is a factor of  $q^s$  better than the bound on insertion correcting codes, despite the fact that any  $s$ -deletion correcting code is an  $s$ -insertion correcting code and vice versa. Note that there is no possible improvement to the insertion channel bound from dividing the inputs into typical and atypical classes.

We extend this bounding strategy to channels that perform both deletions and insertions. We obtain a generalized upper bound that includes Levenshtein's bound as a special case. Recall that Levenshtein's bound was previously known to be tight for one deletion and alphabet size two. The new bound improves upon the previously known bounds whenever the number of deletions is greater than the alphabet size.

The rest of the paper is organized as follows. In Section II, we present some notation and basic results on deletion and insertion channels. In Section III, we construct a class of well-behaved edges in the channel graph. Together with an upper bound on the number of edges in channel graph, the size of this class establishes the asymptotics of the average input degree. In Section IV, we prove a lower bound on the degree of each input vertex and use it to establish our main result: an upper bound on the size of an  $s$ -deletion correcting code.

## II. PRELIMINARIES

### A. Notation

Let  $\mathbb{N}$  be the set of nonnegative integers. Let  $[n]$  be the set of nonnegative integers less than  $n$ ,  $\{0, 1, \dots, n-1\}$ . Let  $2^{[n]}$  be the family of subsets of  $[n]$ . Let  $\binom{[n]}{k}$  be the family of  $k$  element subsets of  $[n]$ . Let  $[q]^n$  be the set of  $q$ -ary strings of length  $n$ . Let  $[q]^*$  be the set of  $q$ -ary strings of all lengths.

We will need the following asymptotic notation: let  $a(n) \sim b(n)$  denote that  $\lim_{n \rightarrow \infty} \frac{a(n)}{b(n)} = 1$  and  $a(n) \lesssim b(n)$  denote

that  $\lim_{n \rightarrow \infty} \frac{a(n)}{b(n)} \leq 1$ . We will use the following asymptotic equality frequently: for fixed  $c$ ,  $\binom{n}{c} \sim \frac{n^c}{c!}$ .

### B. Deletion and insertion channels

We will formalize the problem of correcting deletions and insertions by defining deletion and insertion channels. The  $a$ -deletion  $b$ -insertion channel takes a string of length  $n$ , finds a substring of length  $n-a$ , and outputs a superstring of that substring of length  $n-a+b$ . For strings  $x$  and  $y$ , write  $x < y$  if  $x$  is a substring of  $y$  and define the following sets.

**Definition 1.** For  $x \in [q]^n$ , define  $S_{s,0}(x) = \{z \in [q]^{n-s} : z < x\}$ , the set of substrings of  $x$  that can be produced by  $s$  deletions. Define  $S_{0,s}(x) = \{w \in [q]^{n+s} : w > x\}$ , the set of superstrings of  $x$  that can be produced by  $s$  insertions. Define  $S_{a,b}(x) = \bigcup_{z \in S_{a,0}(x)} S_{0,b}(z)$ .

If  $x$  is the input to an  $n$ -symbol  $a$ -deletion  $b$ -insertion channel,  $S_{a,b}(x)$  is the set of possible outputs.

When two inputs share common outputs they can potentially be confused by the receiver. We are interested in codes that allow the correction of  $s$  deletions. The following sets allow the definition of channels that perform both both types of synchronization errors.

**Definition 2.** A  $q$ -ary  $n$ -symbol  $a$ -deletion  $b$ -insertion correcting code is a set  $C \subset [q]^n$  such that for any two distinct strings  $x, y \in C$ ,  $S_{a,b}(x) \cap S_{a,b}(y)$  is empty.

**Lemma 1.** For  $l, m, n \in \mathbb{N}$  with  $l < m$  and  $l < n$ , let  $x \in [q]^m$  and  $y \in [q]^n$ . Then there exists  $z \in [q]^l$  such that  $x > z$  and  $y > z$  if and only if there exists  $w \in [q]^{m+n-l}$  such that  $w > x$  and  $w > y$ .

**Lemma 2.** For  $a, b, n \in \mathbb{N}$ ,  $x, y \in [q]^n$ ,  $D_{a+b}(x) \cap D_{a+b}(y) = \emptyset$  if and only if  $S_{a,b}(x) \cap S_{a,b}(y) = \emptyset$ . Any  $(a+b)$ -deletion correcting code is also an  $a$ -deletion  $b$ -insertion correcting code.

*Proof:* Suppose there is some  $z \in D_{a+b}(x) \cap D_{a+b}(y)$ . Then there are  $u, v \in [2]^{n-a}$  such that  $x > u > z$  and  $y > v > z$ . The length of  $z$  is  $n-a-b$  so by Lemma 1 there is some  $w$  of length  $(n-a) + (n-a) - (n-a-b) = n-a+b$  such that  $w > u$  and  $w > v$ . Thus  $w$  is in both  $S_{a,b}(x)$  and  $S_{a,b}(y)$ . All of these implications are also true in reverse. ■

**Definition 3.** Let  $B_{q,l,a,b}$  be a bipartite graph with left vertex set  $[q]^{l+a}$  and right vertex set  $[q]^{l+b}$ . Vertices are adjacent if they have a common substring of length  $l$ .

If  $x$  is a left vertex of  $B_{q,l,a,b}$ , then its neighborhood is  $S_{n-l,m-l}(x)$ . This graph completely describes the behavior of the  $n$ -symbol  $(n-l)$ -deletion  $(m-l)$ -insertion channel.

Each  $x \in [q]^{n-s}$  has the same number of superstrings of length  $n$ :

$$|S_{0,s}(x)| = I_{q,s,n}, \quad (1)$$

where

$$I_{q,s,n} = \sum_{i=0}^s \binom{n}{i} (q-1)^i.$$

### III. CONSTRUCTING EDGES

To execute the strategy described in section I-A, we need a lower bound on the degree of a channel inputs. This is a lower bound on the degree of a left vertex of  $B_{q,l,a,b}$ . To obtain this bound, we will first construct a subset of the edges of  $B_{q,l,a,b}$  that is easier to work with than the complete edge set. Our ultimate lower bound on the degree of an input will actually be a lower bound on the number of edges for this subset incident to the input vertex.

One way to get information about the size of a target set  $T$  is to find a construction function  $f : P \rightarrow T$ , where  $P$  is an easily counted parameter set. If  $f$  is injective, then  $|P| = |f(P)|$  and  $|P| \leq |T|$ . We can demonstrate the injectivity of  $f$  with a deconstruction function  $g : T \rightarrow P$  that is a left inverse of  $f$ . This means that  $g(f(p)) = p$  for all  $p \in P$ . If the function  $g$  is given a constructable member of  $A$ , it deconstructs it and recovers the construction parameters. Similarly, if  $f$  is surjective, then we can find an injective  $g : T \rightarrow P$  that is a right inverse of  $f$ , so  $|T| = |g(T)|$  and  $|P| \geq |T|$ . If  $f$  is both injective and surjective, then  $|P| = |T|$ .

In this section we apply these methods to the edge set of  $B_{q,l,a,b}$ . We give an upper bound on the number of edges and discuss why it is difficult to count the edges exactly. We explain our construction of a subset of the edges. Finally we show that the upper and lower bounds match asymptotically.

#### A. An upper bound

By definition, two vertices in  $B_{q,l,a,b}$  are adjacent if they share a substring of length  $l$ . This makes the common substring a natural construction parameter for the edge. We can construct an edge by starting with a string of length  $l$ , performing  $a$  arbitrary insertions to get the left vertex, and performing  $b$  arbitrary insertions to get the right vertex.

**Lemma 3.** *For all  $q, n, a, b \in \mathbb{N}$  with  $s = a + b$ , the number of edges in  $B_{q,l,a,b}$  satisfies*

$$\begin{aligned} |E(B_{q,l,a,b})| &\leq q^l I_{q,a,l+a} I_{q,b,l+b} \\ &\sim q^l \binom{l}{a} (q-1)^a \binom{l}{b} (q-1)^b \\ &\sim q^l \binom{l}{s} (q-1)^s. \end{aligned}$$

*Proof:* There are  $q^l I_{q,a,l+a} I_{q,b,l+b}$  triples  $(z, x, y) \in [q]^l \times [q]^n \times [q]^m$  such that  $z < x$  and  $z < y$ . If  $x \in [q]^n$  and  $y \in [2]^m$  are adjacent, then they have at least one common substring of length  $l$  and appear in at least one triple. ■

This upper bound is not an equality because many pairs of strings  $(x, y) \in [q]^n \times [q]^m$  have multiple common substrings  $z \in [q]^l$ . Pairs of strings with multiple common substrings of length  $l$  fall into two classes. Pairs in the first class have a common substring of length more than  $l$ . Call this string  $w$ . In this case, every substring of length  $l$  of  $w$  is a common substring of the pair. Pairs in the second class have multiple maximum length common substrings. For example, the strings 0101 and 1010 have both 010 and 101 as substrings.

To determine the exact number of edges in  $B_{q,l,a,b}$ , it is necessary to determine the sizes of both classes. The size of the first class can be found easily if the number of edges in  $B_{q,l+i,a-i,b-i}$  is known for all  $i$  up to  $\min(a, b)$ . It is more difficult to characterize the vertex pairs of the second class. Consequently, our lower bound will also not be tight.

#### B. A lower bound

We have constructed every edge at least once by starting with every possible common substring and performing all possible insertions. By using a restricted set of starting substrings and allowed insertions, we will construct each edge at most once. Specifically, we will require that the interval between two insertion points is not alternating.

**Definition 4.** *A string is alternating if some  $u \in [q]$  appears at all even indices, some  $v \in [q]$  appears at all odd indices, and  $u \neq v$ . Let  $A_{q,n}$  be the set of alternating  $q$ -ary strings of length  $n$ .*

The empty string and all strings of length one are trivially alternating. For each length  $n \geq 2$ , each of the  $q$  choices for  $u$  and  $q-1$  choices for  $v$  results in a unique string, so  $|A_{q,n}| = q(q-1)$ . The shortest nonalternating strings have length two, so our restriction prevents two insertions from occurring too close to each other.

To formalize the set of allowable starting substrings, we will need to following definition.

**Definition 5.** *Let the family of compositions with  $t$  dimensions, total multiplicity  $l$ , and minimum multiplicity  $k$  be*

$$M(t, l, k) = \left\{ \mathbf{c} \in (\mathbb{N} \setminus [k])^t \mid \sum_{i \in [t]} c_i = l \right\}.$$

Now we can define the parameter set for the construction function in the lower bound.

**Definition 6.** *For all  $q, l, a, b \in \mathbb{N}$ , let  $s = a + b$  and let*

$$P_{q,l,a,b} = \binom{[s]}{a} \times ([q] \setminus \{0\})^s \times \bigcup_{\mathbf{c} \in M(s+1, l, 0)} \prod_{i=0}^s ([q]^{c_i} \setminus A_{q, c_i})$$

Now we give a summary of the role that the different components of  $P_{q,l,a,b}$  play in construction. The starting substring is specified as  $s+1$  intervals. The length of the  $i$ th interval is  $c_i$ . The total length of the starting substring is  $l$ , so the vector of interval lengths is an element of  $M(s+1, l, 0)$ . An interval cannot be alternating, so it is an element of  $[q]^{c_i} \setminus A_{q, c_i}$ . Each gap between intervals is filled with an inserted symbol in one of the endpoints of edge and nothing in the other endpoint. The subset of the gaps that contains the insertions in the left end point is  $\binom{[s]}{a}$ . The inserted symbols will always differ from the first symbol of the next interval, so there are  $(q-1)$  possibilities for the inserted symbol.

For convenience, we will define our construction and deconstruction functions over larger sets. Start with  $P_{q,l,a,b}$  and drop

the interval nonalternation requirement. Then fix  $s = a + b$  and take the union over  $a \in [s]$  to get

$$2^{[s]} \times ([q] \setminus \{0\})^s \times ([q]^*)^{s+1}.$$

If we represent the subset  $c \in 2^{[s]}$  as a string  $c' \in [2]^s$ , all three terms of the product are lists. To avoid confusion between these bits and the  $q$ -ary symbols that appear everywhere else, we will use the two element set  $\text{LR} = \{\text{LEFT}, \text{RIGHT}\}$  rather than  $[2] = \{0, 1\}$ . To get the final parameter set, we regroup:  $[q]^* \times (\text{LR} \times ([q] \setminus \{0\})) \times [q]^*)^s$ .

Our construction function is Algorithm 1 and our deconstruction function is Algorithm 2. These will treat strings as lists of symbols. We represent the empty list as  $\epsilon$ . The function **HEAD** returns the first symbol of a nonempty list and the function **TAIL** returns everything except the head. The function **LENGTH** returns the number of symbols in the string.

The **CONSTRUCT** function iteratively builds up a pair of strings. **CONSTRUCT** calls **INSERT** once per iteration. The **INSERT** function takes one of the triples described above as an argument and outputs two strings. The two strings are equal to the third component of the triple except that a single symbol has been inserted at the head of one of the outputs.

The **DECONSTRUCT** repeatedly calls **DELETE**. Each **DELETE** undoes the effect of an **INSERT**. **DELETE** takes a pair of strings  $x$  and  $y$  that differ in their first symbol. **DELETE** must pick one of these symbols to be the first symbol of a common substring of the input strings. To decide which first symbol to keep, **DELETE** calls **MATCH** twice. The **MATCH** function takes two strings  $x$  and  $y$ , finds their longest common prefix, and outputs the prefix and the two corresponding suffixes. **DELETE** calls **MATCH** on  $(\text{TAIL}(x), y)$  and  $(x, \text{TAIL}(y))$  and then performs the deletion that resulted in a longer common prefix. The information about the deletion and prefix become a triple. **DELETE** returns this triple along with two suffices from the match.

### C. Deconstructability

Now we will show that **DECODE** is a left inverse of **ENCODE**. The first step is to look at the inner functions: **INSERT** and **DELETE**.

**Lemma 4.** For  $lr \in \text{LR}$ ,  $\delta \in [q] \setminus \{0\}$ , and  $w \in [q]^m \setminus A_{q,m}$ , let  $(x, y) = \text{INSERT}(lr, \delta, w)$ . Let  $u$  and  $v$  be arbitrary  $q$ -ary strings with different first symbols. Then  $\text{DELETE}(x : u, y : v) = ((lr, \delta, w), u, v)$ .

*Proof:* Let  $w = (w_0, w_1, \dots, w_{m-1})$ . Without loss of generality let  $lr = \text{LEFT}$ , so  $x = (w_0 + \delta) : b$  and  $y = b$ . First, **DELETE** computes  $g = (w_0 + \delta) - w_0 = \delta$ . Next, it evaluates  $\text{MATCH}(w : u, w : v)$  and obtain  $(w, u, v)$  because  $u_0 \neq v_0$ . Thus the length of the first match is  $\text{LENGTH}(b) = m$ . Second, it evaluates  $\text{MATCH}((w_0 + \delta) : w : u, w : v)$ . If the length of the second match is at least  $m - 1$ , then  $w_0 + \delta = w_1$  and  $w_i = w_{i+2}$  for  $0 \leq i \leq m - 3$ . This would make  $w$  alternating, so the length of the second match is at most  $m - 2$ . The first match is longer than the second, so the first branch of the if

---

### Algorithm 1 Construct an edge

---

```

CONSTRUCT :  $[q]^* \times (\text{LR} \times ([q] \setminus \{0\})) \times [q]^*)^s \rightarrow [q]^* \times [q]^*$ 
CONSTRUCT( $z_0, t$ )
   $x \leftarrow z_0$ 
   $y \leftarrow z_0$ 
  while  $z \neq \epsilon$  do
     $(u, v) \leftarrow \text{INSERT}(\text{HEAD}(t))$ 
     $t \leftarrow \text{TAIL}(t)$ 
     $x \leftarrow x : u$ 
     $y \leftarrow y : v$ 
  end while
  return  $(x, y)$ 

INSERT :  $\text{LR} \times ([q] \setminus \{0\}) \times [q]^* \rightarrow [q]^* \times [q]^*$ 
INSERT( $lr, \delta, w$ )
   $w' \leftarrow (\delta + \text{HEAD}(w)) : w$ 
  if  $lr = \text{LEFT}$  then
    return  $(w', w)$ 
  else
    return  $(w, w')$ 
  end if

```

---

statement is taken and the function returns  $((\text{LEFT}, \delta, w), u, v)$ . ■

**Lemma 5.** For  $0 \leq i \leq s$  let  $w_i$  be  $q$ -ary string that is not alternating. For  $1 \leq i \leq s$  let  $lr_i \in \{\text{LEFT}, \text{RIGHT}\}$ , let  $\delta_i \in [q] \setminus \{0\}$ , and let  $t_i = (lr_i, \delta_i, w_i)$ . Let  $t = (t_1, \dots, t_s)$ . Then  $\text{DECONSTRUCT}(\text{CONSTRUCT}(w_0, t)) = (w_0, t)$ .

*Proof:* The strings output by **CONSTRUCT** are the concatenation of  $s + 1$  intervals. The first pair of intervals are equal to  $z_0$ . The remaining  $s$  pairs are each produced by one call to **INSERT**, so their initial symbols differ. Consequently, the initial call to **MATCH** in **DECONSTRUCT** finds  $z_0$ . After that, the conditions of Lemma 4 are met. Each call to **DELETE** recovers the input to one **INSERT** operation and preserves the conditions of Lemma 4. ■

**Lemma 6.** For all  $q, l, a, b \in \mathbb{N}$ , there are functions  $\text{CONSTRUCT} : P_{q,l,a,b} \rightarrow E(B_{q,l,a,b})$  and  $\text{DECONSTRUCT} : E(B_{q,l,a,b}) \rightarrow P_{q,l,a,b}$  such that  $\text{DECONSTRUCT}(\text{CONSTRUCT}(p)) = p$  for all  $p \in P_{q,l,a,b}$ .

*Proof:* From Lemma 5, **DECONSTRUCT** is a right inverse of **CONSTRUCT**. The discussion at the beginning of this section describes the bijection between  $P_{q,l,a,b}$  and the input to **CONSTRUCT**. It is easy to check that the strings produced by **CONSTRUCT** have lengths  $l + a$  and  $l + b$  and have a common substring of length  $l$ . ■

**Lemma 7.** For fixed  $q, a, b \in \mathbb{N}$ ,  $|P_{q,l,a,b}| \gtrsim q^l \binom{l}{s} \binom{s}{a} (q - 1)^s$ .

*Proof:* First,  $\left| \binom{[s]}{a} \right| = \binom{s}{a}$  and  $|([q] \setminus \{0\})^s| = (q - 1)^s$ . For  $c_i \geq 2$ ,  $|[q]^{c_i} \setminus A_{q,c_i}| = q^{c_i} - q(q - 1)$ . The number of

---

**Algorithm 2** Deconstruct an edge

---

```

DECONSTRUCT :  $[q]^* \times [q]^* \rightarrow [q]^* \times (\text{LR} \times ([q] \setminus \{0\}) \times [q]^*)^s$ 
DECONSTRUCT( $x, y$ )
   $(z_0, x, y) \leftarrow \text{MATCH}(x, y)$ 
   $t \leftarrow \epsilon$ 
  while  $x \neq \epsilon \wedge y \neq \epsilon$  do
     $(w, x, y) \leftarrow \text{DELETE}(x, y)$ 
     $t \leftarrow t : w$ 
  end while
  assert  $x = \epsilon \wedge y = \epsilon$ 
  return  $(z_0, t)$ 

DELETE :  $[q]^* \times [q]^* \rightarrow (\text{LR} \times ([q] \setminus \{0\}) \times [q]^*) \times [q]^* \times [q]^*$ 
DELETE( $x, y$ )
   $g = \text{HEAD}(x) - \text{HEAD}(y)$ 
   $(a, b, c) \leftarrow \text{MATCH}(\text{TAIL}(x), y)$ 
   $(d, e, f) \leftarrow \text{MATCH}(x, \text{TAIL}(y))$ 
  assert  $\text{LENGTH}(a) \neq \text{LENGTH}(d)$ 
  if  $\text{LENGTH}(a) > \text{LENGTH}(d)$  then
    return  $((\text{LEFT}, g, a), b, c)$ 
  else
    return  $((\text{RIGHT}, (-g), d), e, f)$ 
  end if

MATCH :  $[q]^i \times [q]^j \rightarrow [q]^k \times [q]^{i-k} \times [q]^{j-k}$ 
MATCH( $x, y$ )
   $w \leftarrow \epsilon$ 
  while  $x \neq \epsilon \wedge y \neq \epsilon \wedge \text{HEAD}(x) = \text{HEAD}(y)$  do
     $w \leftarrow w : \text{HEAD}(x)$ 
     $x \leftarrow \text{TAIL}(x)$ 
     $y \leftarrow \text{TAIL}(y)$ 
  end while
  return  $(w, x, y)$ 

```

---

sequences of strings is

$$\begin{aligned}
& \sum_{c \in M(s+1, l, 2)} \prod_{i=0}^s (q^{c_i} - q(q-1)) \\
& \geq q^l \sum_{c \in M(s+1, l, 2)} \prod_{i=0}^s (1 - q^{2-c_i}) \\
& \geq q^l \sum_{c \in M(s+1, l, 2+\log_q l)} \prod_{i=0}^s (1 - q^{-\log_q l})
\end{aligned}$$

Because  $|M(t, l, k)| = \binom{l+(1-k)t-1}{t-1}$ , this equals

$$q^l \binom{l - (1 + \log_q l)(s+1) - 1}{s} (1 - l^{-1})^{s+1} \sim q^l \binom{l}{s}.$$

Our bounds establish the asymptotic growth of the number of edges.

**Theorem 1.** *The number of edges in  $B_{q,l,a,b}$  satisfies  $|E(B_{q,l,a,b})| \sim q^l \binom{l}{s} \binom{s}{a} (q-1)^s$ . The average of  $S_{a,b}(x)$  over all  $x \in [q]^n$  is asymptotic to  $\binom{n}{s} \binom{s}{a} (q-1)^s q^{-a}$ .*

*Proof:* The first claim follows immediately from Lemmas 3, 6, and 7. For  $x \in [q]^n$ , the set  $S_{a,b}(x)$  is the neighborhood of  $x$  in  $B_{q,n-a,a,b}$ . Each edge involves exactly one of the  $q^n$  left vertices and  $\binom{n-a}{a} \sim \binom{n}{a}$ . ■

Now we can conclude that most edges are constructable by our method. This is a necessary condition for the asymptotic tightness of our ultimate lower bound on input degree.

#### IV. BOUNDS ON INPUT DEGREE AND CODE SIZE

**Lemma 8.** *Let  $x \in [q]^n$  be a string with  $r$  runs. Let  $c$  be the length the longest alternating subinterval of  $x$ . Then  $|S_{a,b}(x)|$ , the number of unique strings that can be produced from  $x$  by  $a$  deletions and  $b$  insertions, is at least*

$$\binom{r-a-2-(a+1)c}{a} \binom{n-2a-1-(2a+b+1)c}{b} (q-1)^b.$$

*Proof:* Given a string  $x \in [q]^n$ , we identify a subset of  $P_{q,n-a,a,b}$ . Application of CONSTRUCT from Lemma 6 to any member of this subset produces an edge with left endpoint  $x$ . By Lemma 6, each left endpoint is produced at most once.

We select  $a$  symbols of  $x$  for deletion, select  $b$  spaces between symbols for insertion, and specify the  $b$  new symbols. The selected symbols and spaces partition  $x$  into  $s+1$  intervals. To ensure that none of these intervals are alternating, we will require that all of the intervals contain at least  $c+1$  symbols.

Deleting any of the symbols in a run has the same effect, so we will select symbols that occur at the left ends of runs. We cannot select the last symbol of the string, so there are  $r-1$  symbols to choose from. To ensure that there are at least  $c+1$  symbols between consecutive deleted symbols, we will require that there be  $c+1$  deletable symbols. There are  $\binom{r-1-(a+1)(c+1)}{a}$  ways to pick the symbols for deletion.

There are  $m-1$  potential spaces to make an insertion. Insertions cannot be done in the  $c+1$  spaces before and after a deleted symbol. In the worst case, all of these forbidden spaces are distinct, leaving  $n+1-2a(c+1)$  spaces to choose from. There must be  $c+1$  symbols and  $c$  spaces between any two consecutive chosen spaces. Thus there are always at least  $\binom{n-1-2a(c+1)-(b+1)c}{b}$  ways to pick the spaces.

Finally, for each of the  $b$  insertion points, we must specify the inserted symbol. To do this, for each insertion point we pick  $\delta \in [q] \setminus \{0\}$  and make the new symbol equal to  $\delta$  plus its successor. Thus, there are  $(q-1)^b$  choices for this step.

Slight rearrangement gives the claimed result. ■

To apply Lemma 8 to a string, we need two statistics of that string: the number of runs and the length of the longest alternating subinterval. The next two lemmas concern the distributions of these statistics.

**Lemma 9.** *The number of  $q$ -ary strings of length  $n$  with an alternating subinterval of length at least  $c$  is at most  $(n-c+1)q^{n-c+1}(q-1)$ .*

*Proof:* A string of length  $n$  contains  $n-c+1$  intervals of length  $c$ . If some subinterval of length at least  $c$  is alternating, at least one of subintervals of length exactly  $c$  is alternating.

There are  $q(q-1)$  choices for the alternating interval and  $q^{n-c}$  choices for the remaining symbols. ■

**Lemma 10.** *The number of  $q$ -ary strings of length  $n$  with  $\left(\frac{q-1}{q} - \epsilon\right)(n-1) + 1$  or fewer runs is at most  $q^n e^{-2(n-1)\epsilon^2}$ .*

*Proof:* For  $x \in [q]^n$ , let  $x' \in [q]^{n-1}$  be the string of first differences of  $x$ . That is, let  $x'_i = x_{i+1} - x_i \bmod q$ . If  $x$  has  $r$  runs, then  $x'_i$  is nonzero at the  $r-1$  boundaries between runs. Thus there are  $q \binom{n-1}{r-1} (q-1)^{r-1}$  strings with exactly  $r$  runs. Now we can apply the following Chernoff inequality:

$$\sum_{i=0}^{\left(\frac{q-1}{q} - \epsilon\right)(n-1)} \binom{n-1}{i} (q-1)^i \leq q^{n-1} e^{-2(n-1)\epsilon^2}.$$

Now we have all of the ingredients required to execute the strategy described in Section I-A.

**Theorem 2.** *For fixed  $q, a, b \in \mathbb{N}$  and  $s = a + b$ , the number of codewords in an  $n$ -symbol  $q$ -ary  $a$ -deletion  $b$ -insertion correcting code is asymptotically at most*

$$\frac{q^{n+b}}{(q-1)^s \binom{n}{s} \binom{s}{b}}.$$

*Proof:* We divide strings into three classes: typical strings, strings with a long alternating subinterval, and strings with few runs. Call these classes  $C_1$ ,  $C_2$ , and  $C_3$  respectively.

A string will fall into  $C_2$  if it has an alternating subinterval of length at least  $c$ . If we let  $c = (s+2) \log_q n$ , then by Lemma 9 we have  $|C_2| < nq^{n-c+1}(q-1) = n^{-(s+1)} q^{n+1}(q-1)$  which is  $O(q^n/n^{s+1})$ .

The average number of runs is  $\frac{q-1}{q}(n-1) + 1$ . A string will fall in the third class if it has at most  $\left(\frac{q-1}{q} - \epsilon\right)(n-1) + 1$  runs. If we let  $\epsilon = \sqrt{\frac{(s+1) \log n}{2(n-1)}}$ , then by Lemma 10 we have

$$|C_3| \leq q^n e^{-2(n-1)\epsilon^2} = q^n e^{-(s+1) \log n} = q^n / n^{s+1}.$$

For fixed  $s$ , this  $\epsilon$  is  $o(1)$ , so  $\left(\frac{q-1}{q} - \epsilon\right)(n-1) + 1 \sim \frac{(q-1)n}{q}$ .

Now we can apply Lemma 8 to lower bound the degree of the typical strings. As before, let  $s = a + b$ . The first multiplicative term in the lower bound is asymptotic to

$$\begin{aligned} \binom{\frac{q-1}{q}n - (a+1)(s+2) \log_q n}{a} &\sim \binom{\frac{q-1}{q}n}{a} \\ &\sim \left(\frac{q-1}{q}\right)^a \binom{n}{a}. \end{aligned}$$

The second term is asymptotic to

$$\binom{n - (2a+b+1)(s+2) \log_q n}{b} \sim \binom{n}{b}.$$

Thus

$$\begin{aligned} \min_{x \in C_1} |S_{a,b}(x)| &\gtrsim \left(\frac{q-1}{q}\right)^a \binom{n}{a} \binom{n}{b} (q-1)^b \\ &\sim \frac{(q-1)^s}{q^a} \binom{n}{s} \binom{s}{b}. \end{aligned}$$

There are  $n - a + b$  possible outputs, so the number of codewords is at most

$$\begin{aligned} &\frac{q^{n-a+b}}{\min_{x \in C_1} |S_{a,b}(x)|} + |C_2| + |C_3| \\ &\lesssim \frac{q^{n-a+b}}{\frac{(q-1)^s}{q^a} \binom{n}{s} \binom{s}{b}} + \frac{q^{n+1}(q-1)}{n^{s+1}} + \frac{q^n}{n^{s+1}} \\ &\sim \frac{q^{n+b}}{(q-1)^s \binom{n}{s} \binom{s}{b}} \end{aligned}$$

By setting  $b$  to zero we recover Levenshtein's upper bound. The generalized bound offers an improvement whenever 1 is a better choice for  $b$  than 0. This occurs whenever  $s > q$ .

The best bound is achieved when  $\frac{q^b}{\binom{s}{b}}$  is minimized. This occurs when  $b = \lceil \frac{s-q}{q+1} \rceil$ . If we pick this value for  $b$ , there is some  $c \in [q+1]$  such that  $b = \frac{s-q+c}{q+1}$  and  $s = b(q+1) + q - c$ . Then the improvement over Levenshtein's bound is

$$\frac{\binom{s}{b}}{q^b} = \frac{1}{b!q^b} \prod_{i=0}^{b-1} (bq + b + q - c - i) \geq \frac{(bq)^b}{b!q^b} = \frac{b^b}{b!} \geq \frac{e^{b-1}}{\sqrt{b}}$$

The first inequality is true because  $c \leq q$  and  $i < b$ . The second inequality comes from Stirling's approximation.

#### ACKNOWLEDGMENT

This work was supported in part by AFOSR under grants FA 9550-11-1-0016 and FA 9550-10-1-0573; and by NSF grant CCF 10-54937 CAR.

#### REFERENCES

- [1] L. Calabi and W. E. Hartnett, "Some general results of coding theory with applications to the study of codes for the correction of synchronization errors\*," *Information and Control*, vol. 15, no. 3, p. 235249, 1969.
- [2] D. Cullina, A. Kulkarni, and N. Kiyavash, "A coloring approach to constructing deletion correcting codes from constant weight subgraphs," in *2012 IEEE International Symposium on Information Theory Proceedings (ISIT)*, July 2012, pp. 513–517.
- [3] D. Hirschberg, "Bounds on the number of string subsequences," in *Combinatorial Pattern Matching*, 1999, p. 115122.
- [4] A. A. Kulkarni and N. Kiyavash, "Non-asymptotic upper bounds for deletion correcting codes," *IEEE Transactions on Information Theory*, accepted, 2012. [Online]. Available: <http://arxiv.org/abs/1211.3128>
- [5] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, 1966, p. 707710.
- [6] —, "Elements of coding theory," *Diskretnaya matematika i matematicheskie voprosy kibernetiki*, p. 207305, 1974.
- [7] —, "Bounds for deletion/insertion correcting codes," in *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, 2002, p. 370.
- [8] Y. Liron and M. Langberg, "A characterization of the number of subsequences obtained via the deletion channel," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, 2012, p. 503507.
- [9] T. G. Swart and H. C. Ferreira, "A note on double insertion/deletion correcting codes," *Information Theory, IEEE Transactions on*, vol. 49, no. 1, p. 269273, 2003.
- [10] R. Varshamov, "On an arithmetic function with an application in the theory of coding," *Doklady Akademii nauk SSSR*, vol. 161, pp. 540–543, 1965.
- [11] R. Varshamov and G. Tenengolts, "Codes which correct single asymmetric errors," *Avtomatika i Telemekhanika*, vol. 26, p. 288292, 1965.